



TITLE:

並列プログラム図式に関するいくつかの性質 (計算機科学の数学的基礎)

AUTHOR(S):

山下, 雅史; 稲垣, 康善; 本多, 波雄

CITATION:

山下, 雅史 ...[et al]. 並列プログラム図式に関するいくつかの性質 (計算機科学の数学的基礎). 数理解析研究所講究録 1978, 322: 99-116

ISSUE DATE:

1978-03

URL:

<http://hdl.handle.net/2433/104030>

RIGHT:

並列プログラム関式に関するいくつかの性質

名 大・工	山下 雅史
三重大・工	稲垣 康善
名 大・工	本多 波雄

1. はじめに

並列プログラム, OS等の研究が進むにつれて, 並列処理プログラムを記述し, 解析するための, いくつかのモデルが提案され試みられてきた。しかし, それらのいずれも, つぎの諸点に対して十分に答えているとは思えない。

(1) 実際にはインプリメントされている, 例へば"Concurrent-PASCAL"の様な高級言語との関係を容易に示すことができない。

(2) 制御構造, 及び, プロセスの記述を含むプログラム全体の記述が十分に可能であり, 且つ形式的な取扱いにも耐えうる。

(3) 各プロセスのスケジューリング方式とも記述することが可能。

本報告では、以上の諸点に十分答えるモデルを提案し、デッドロック、相互排他という並列処理プログラムの基本的な問題を考察する。

2. 並列プログラム四式 (PPS) の定義と実行

(定義 1) PPS は以下で定義される。

$$PPS = (\text{control part}, \text{program part})$$

$$\text{control part} = (\text{scheduler}_1, \dots, \text{scheduler}_m)$$

$$\text{scheduler} = (K, \Sigma, \delta, s_0)$$

ここに、 K : 状態の有限集合、 Σ : 入力シンボルの有限集合、 $\delta: K \times \Sigma \rightarrow K$ の関数、 $s_0 \in K$: 初期状態である。

$$\text{program part} = (\text{process}_1, \dots, \text{process}_n)$$

$$\text{process} = (S_1, \dots, S_p)$$

$$S_i = i. \text{ statement}$$

$$\text{statement} = \text{assignment statement} / \text{test statement} / \text{halt statement} / \text{communication statement}$$

ここに、assignment statement: $Y \leftarrow f(Y)$, test statement: if P then goto i else goto j , halt statement: halt, communication statement: $c \in \Sigma$ である。 ■

PPS の動く仮想計算機 (VM) には、2 つの関数 exec と next が備わっている。 next は次の時点で実行するプロセスを定め

る関数であり, $exec$ は VM による計算を定める関数である。
次に PPS の計算状況を定義し, その推移により, PPS の実行
を定義する。

(定義2) pps の計算状況は4項組 (NP, LC, Q, Y) である。

ここに, NP : 実行されるプロセス番号 ($1 \leq NP \leq n$; n はプロ
セス数), LC : 各プロセスのロケーション・カウンタの値
を示すベクトル ($LC \triangleq (loc_1, \dots, loc_n)$), Q : 各スケジュー
ーラの状態を示すベクトル ($Q \triangleq (q_1, \dots, q_m)$; m はスケ
ジューラ数), Y : プログラム変数の値を示すベクトル (Y
 $\triangleq (y_1, \dots, y_k)$) である。 ■

(定義3) 計算状況の推移 $(NP_t, LC_t, Q_t, Y_t) \rightarrow (NP_{t+1}, LC_{t+1},$
 $Q_{t+1}, Y_{t+1})$ を以下で定義する。

$$(LC_{t+1}, Q_{t+1}, Y_{t+1}) = exec(stat_{NP_t}, loc_{NP_t}, LC_t, Q_t, Y_t)$$

($stat_{ij}$ はプロセス i ロケーション j のステートメント)。

関数 $exec: (LC', Q', Y') = exec(stat_{ij}, LC, Q, Y)$ は,

$stat_{ij}$ が $Y \leftarrow f(Y)$ なら $LC' = LC_{loc_{i+1}}^i$; $Y' = f(Y)$, $stat_{ij}$

が if P then goto k else goto l なら $LC' = \text{if } P \text{ then } LC_k^i \text{ else}$

LC_l^i , $stat_{ij}$ が $c \in \mathbb{Z}_k$ なら if $\delta_k(q_k, c) = q'_k$ then ($LC' =$

$LC_{loc_{i+1}}^i$; $Q' = Q_{q'_k}^k$), $stat_{ij}$ が halt なら $LC' = LC_k^i$

(A_j^i はベクトル A の i 番目の要素に値 j を入れることを示
す)。

$NP_{t+1} = \text{next}(NP_t, LC_{t+1}, Q_{t+1}, Y_{t+1})$ ($\text{loc} NP_{t+1} \neq h$) である。

next , loc , h は普通の方法でトから拡張して定義されるものである。 ■

(定義4) ppsの実行は計算状況の推移列

$(NP_0, LC_0, Q_0, Y_0) \vdash (NP_1, LC_1, Q_1, Y_1) \vdash \dots$ である。ここに、

$LC_0 \triangleq (1, \dots, 1)$, $Q_0 \triangleq (q_0^1, \dots, q_0^m)$ (q_0^i はスケジューラ
の初期状態), $Y_0 \triangleq (\omega, \dots, \omega)$ (ω は未定義記号), NP_0 :
初期プロセス番号である。

$LC_h \triangleq (h, \dots, h)$, $NP_h \triangleq (NP, LC_h, Q, Y)$ とすると、計算
状況 $\forall (NP_h, LC_h, Q, Y)$ に到達すると pps は停止する。 ■

pps の例を図1に示す。以降で考察するのは図1の pps に対する
デッドロック並びに相互排他問題である。

3. ppsのデッドロック並びに相互排他問題

3.1 デッドロックと相互排他

(定義5) ppsのある計算状況 (NP, LC, Q, Y) が任意の $n \geq 0$
に対して $(NP, LC, Q, Y) \not\vdash (NP', LC, Q, Y)$ となるとき、p
ps はデッドロック状態にあるという。 ■

本節で述べた pps 及びその実行の定義より、関数 exec は、
 $\text{communication statement}$ (com. stat.) 以外のステート

<pre> process₁ 1. $\bar{y}_1 \leftarrow \bar{x}_1$ 2. if $P_1(\bar{y}_1)$ then goto 3 else goto $3(\alpha+1)+1$ 3. a_1 4. $\bar{y}_s \leftarrow f_{11}(\bar{y}_1, \bar{y}_s)$ 5. $\bar{y}_1 \leftarrow f_{21}(\bar{y}_1, \bar{y}_s)$: 3α. a_α 3$\alpha+1$. $\bar{y}_s \leftarrow f_{1\alpha}(\bar{y}_1, \bar{y}_s)$ 3$\alpha+2$. $\bar{y}_1 \leftarrow f_{2\alpha}(\bar{y}_1, \bar{y}_s)$ 3($\alpha+1$). goto 2 3($\alpha+1$)+1. halt </pre>	<pre> process₂ 1. $\bar{y}_2 \leftarrow \bar{x}_2$ 2. if $P_2(\bar{y}_2)$ then goto 3 else goto $3(\beta+1)+1$ 3. b_1 4. $\bar{y}_s \leftarrow g_{11}(\bar{y}_2, \bar{y}_s)$ 5. $\bar{y}_2 \leftarrow g_{21}(\bar{y}_2, \bar{y}_s)$: 3β. b_β 3$\beta+1$. $\bar{y}_s \leftarrow g_{1\beta}(\bar{y}_2, \bar{y}_s)$ 3$\beta+2$. $\bar{y}_2 \leftarrow g_{2\beta}(\bar{y}_2, \bar{y}_s)$ 3($\beta+1$). goto 2 3($\beta+1$)+1. halt </pre>
---	--

scheduler $\mathcal{S} = (K, \Sigma, \delta, q_0)$ $\Sigma = \{a_i\text{'s}, b_j\text{'s}\}$

(図1) 本稿で考察する PPS の例

(注) goto i if true then goto i else goto i
の省略形として使用する。

\times と \uparrow に対しては、LC の値を変える。したがって、デッドロックは com.stat., scheduler, 及び関数 next のみに依存する現象であることがわかる。デッドロックは next に依存して生じるので、実際ある種の next に対しては、ナニセニスな状態でデッドロックする。そこで本報告では、next は次の意味で公平 (fair) であるとする。

(定義6) 関数 next が公平 (fair) であるとは、任意の計算状況 (NP, LC, Q, Y) ($\text{stat } NP_{loc} NP \neq \text{halt}$) に対して、ある $n \geq 1$ が存在して $(NP, LC, Q, Y) \models (NP, LC', Q', Y')$ となることである。

(補題1) $next$ が公平であるとき, PPSがデッドロック状態にあることと, PPSの計算状況 (NP, LC, Q, Y) が任意の, $loc_i \neq h$ である i に対して $exec(stat_i loc_i, LC, Q, Y) = (LC, Q, Y)$ であることとは同値である.

(証明) PPSの実行の定義, 公平の定義より明らか. ■

特定のいくつかのステートメント又はステートメントの列が同時に実行されないとき, それらは互いに相互排他されているという. 2節で述べたいくつかの定義より明らかのように, PPSでは各ステートメント単位での相互排他は実現されている. 本報告では少し大きな範囲での相互排他を考える. すなわち, 2つの $com.stat.$ には含まれその間には $com.stat.$ を含まない一連のステートメントの列をブロックと呼び, そのようなブロック毎の相互排他を考える. ブロックは1つのプロセスに含まれるステートメントの集合であるので, B_{iI} (i : 自然数; I : 自然数の有限部分集合) で $process_i$ の I の中の数をロケーションとして持つステートメントよりなるブロックを表現することにする.

(定義7) $B_{iI_1}, \dots, B_{iI_n}$ の中の任意の2つのブロック B_{iI}, B_{jJ} について, 任意に $k \in I, l \in J$ をとるとき, PPSの実行の任意の計算状況 (NP, LC, Q, Y) において, $loc_i = k \wedge loc_j = l$ となることのないとき, $B_{iI_1}, \dots, B_{iI_n}$ は互いに相互排他

されているという。 ─

相互排他の問題を扱う場合にもデッドロック問題のときと同様に関数 $next$ は公平であるとする。

3.2 問題と解法

図1はプロセス間通信を含む Concurrent-PASCAL プログラムに対して現われる PPS の一例である。しかも、その $process_1$ のループ回数と $process_2$ のループ回数の比が $n: f(n)$ (f : 多項式) で表現できる場合ばかりではない。以下では、このような場合に PPS がデッドロックに陥る可能性があるか否か、そして相互排他が指定したブロックの集合に対して実現されているか否かは決定可能であることを示す。

(定義8) n -テープ有限オートマトン (n -TFA) とは5項組 $(K, \Sigma, \delta, q_0, F)$ である。ここに、 K : 状態の有限集合, Σ : 入力シンボルの有限集合でその中の特殊な元 $\$$ をエンドマークとして使用する, $\delta: K \times \Sigma \times \Sigma \rightarrow 2^{K \times \{0,1\} \times \{0,1\}}$, $q_0 \in K$: 初期状態, $F \subseteq K$: 最終状態の集合, である。 ─

(定義9) n -TFA \mathcal{A} の計算状況は3項組 (q, r, s) である。

ここに、 q : 現在の状態, r : \mathcal{A} の1テープの端からヘッドのある位置までの距離, s : \mathcal{A} の n テープの端からヘッドのある位置までの距離, である。 ─

(定義10) 計算状況の推移は以下で定義される.

$\delta(q, (a_r, b_s)) \ni (q', (d_1, d_2))$ (a_r は 1 テーポの r 番目の
シンボル, b_s は 2 テーポの s 番目のシンボル, $d_1, d_2 \in$
 $\{0, 1\}$) のとき, $(q, r, s) \vdash (q', r+d_1, s+d_2)$ である.

$\vdash, \vdash^*, \vdash^+$ は普通の方法で \vdash から拡張されて定義されるもの
とする. ■

(定義11) $a_1 \dots a_{r-1}, b_1 \dots b_{s-1} \in (\Sigma - \{\$ \})^*$, $T_1 = a_1 \dots a_{r-1} \$$, $T_2 = b_1$
 $\dots b_{s-1} \$$ とすると, 計算状況の列 $(q_0, 1, 1) \vdash^* (q_F, \ell, m)$
($q_F \in F$, $\ell \leq r$, $m \leq s$) が存在すれば, σ は語 (T_1, T_2) を受理
するといふ. ($\forall q_F \in F$ に対する δ は常に \emptyset であり, δ の値は
($q_F, (0, 0)$) であるとする.) ■

(補題2) $W_D = \{((a_1 \dots a_\ell)^n \$, (b_1 \dots b_\ell)^{+n} \$)\} \quad (n \geq 1)$ とする.

このとき, 図1の PPS パテックブロックある可能性を持たない
ことと $\mathcal{L}(\sigma_D) \cap W_D = \emptyset$ とが同値となる 2-TFA σ_D
が存在する.

(略証) 図1の scheduler $\mathcal{S} = (K, \Sigma, \delta, q_0)$ に対して, 2-T
FA $\sigma_D = (K', \Sigma', \delta', q'_0, F')$ を以下のように構成する.

• $K' = K \cup \{q_F\}$, $\Sigma' = \Sigma \cup \{\$ \}$, $q'_0 = q_0$, $F' = \{q_F\}$,

$\delta(q, a) = q' \quad (a \in \Sigma)$ のとき, $\forall b \in \Sigma'$ に対して $\delta'(q, (a, b)) \ni$
($q', (1, 0)$) 且 $\delta'(q, (b, a)) \ni (q', (0, 1))$, $\delta(q, a) = \emptyset \quad (a \in \Sigma)$ の
とき, $\delta'(q, (a, \$)) = \delta'(q, (\$, a)) \ni (q_F, (0, 0))$, $\delta(q, a) = \delta$

$(q, b) = \phi \ (a, b \in \Sigma) \text{ かつ } \delta'(q, (a, b)) = \delta'(q, (b, a)) \Rightarrow (q_F, (0, 0))$

このとき, \mathcal{S}_M は PPS の制御をシミュレートして, PPS のデッドロックに入るとき, そのときに限り受理に至る. ■

$M_1 = \{B_1^1, \dots, B_{\lambda_1}^1\}, \dots, M_\ell = \{B_1^\ell, \dots, B_{\lambda_\ell}^\ell\}$ をそれぞれ互いに相互排他が期待されるブロックの集合とする. ここに任意の j, k に対して, $M_j \cap M_k = \phi$ である. スtring $S_1 = a_1 \dots a_\alpha$ に対して, その中の a_j と a_{j+1} の間にブロックが M_i に含まれるとき, シンボル M_i を a_j と a_{j+1} の間に挿入した string を S_1' とする. $S_2 = b_1 \dots b_\beta$ に対して同様の操作を行い S_2' とする. $W_M = \{(S_1')^n \$, (S_2')^{f(n)} \$\} \ (n \geq 1)$ とする.

(補題3) 図1の PPS のデッドロックする可能性がなく, 且つ M_1, \dots, M_ℓ の各集合に含まれるブロック間の相互排他が実現されていることと $\mathcal{L}(\mathcal{S}_M) \cap W_M = \phi$ とが同値となるような 2-TFA \mathcal{S}_M が存在する.

(略証) 図1の scheduler $\mathcal{A} = (K, \Sigma, \delta, q_0)$ に対して, 2-TFA $\mathcal{S}_M = (K'', \Sigma'', \delta'', q_0'', F'')$ を以下のように構成する.

• $K'' = K \cup \{q_F\}$, $\Sigma'' = \Sigma \cup \{\phi\} \cup \{M_1, \dots, M_\ell\}$, $q_0'' = q_0$, $F'' \ni q_F$
 $\delta(q, a) = q' \ (a \in \Sigma)$ のとき, $\forall b \in \Sigma''$ に対して $\delta''(q, (a, b)) \Rightarrow (q', (1, 0))$ 且つ $\delta''(q, (b, a)) \Rightarrow (q', (0, 1))$, $\delta(q, a) = \phi \ (a \in \Sigma)$ かつ $\exists, \delta''(q, (a, \phi)) = \delta''(q, (\phi, a)) \Rightarrow (q_F, (0, 0))$, $\delta(q, a) = \delta(q, b) = \phi$
 $(a, b \in \Sigma)$ のとき, $\delta''(q, (a, b)) = \delta''(q, (b, a)) \Rightarrow (q_F, (0, 0))$

$\forall i, \forall q$ に対して、 $b \in \Sigma^* - \{M_i\}$ とすると、 $\delta''(q, (M_i, b)) \Rightarrow (q, (1, 0))$ 且 $\delta''(q, (b, M_i)) \Rightarrow (q, (0, 1))$ 且、 $\delta''(q, (M_i, M_i)) \Rightarrow (q_F, (0, 0))$. \blacksquare

補題 2.3 に より、ここでの問題は 2-TFA と語の集合 $W = \{(a_1 \dots a_\alpha)^n \$, (b_1 \dots b_\beta)^{f(n)} \$\} (n \geq 1)$ に対して、 $\mathcal{L}(\mathcal{G}) \cap W = \emptyset$ かの決定問題に帰着された。

2-TFA と W に対して、2-TFA $\mathcal{G}' = (K', \Sigma', \delta', q_0', F')$ ($\Sigma' = \{a, b, \$\}$) を以下の通りに構成する。

- $K - F \ni q_i a$ とし、 $K' \ni q_{ijk} (1 \leq j \leq \alpha, 1 \leq k \leq \beta)$
- $\Sigma' = \{a, b, \$\}$, $q_0' = q_{011}$, $F' = \{q_F\}$
- $\delta(q_i, (a_j, b_k)) \Rightarrow (q_{\ell}, (\vartheta_1, \vartheta_2))$ ($\vartheta_1, \vartheta_2 \in \{0, 1\}$, $q_{\ell} \notin F$) のとき、
 $\delta'(q_{ijk}, (a, b)) \Rightarrow (q_{\ell j'k'}, (\vartheta_1, \vartheta_2))$ ($j' = \text{if } (\vartheta_1 = 1) \text{ then } (j+1 \bmod \alpha) + 1 \text{ else } j, k' = \text{if } (\vartheta_2 = 1) \text{ then } (k+1 \bmod \beta) + 1 \text{ else } k$)
- $\delta(q_i, (a_j, b_k)) \Rightarrow (q_F, (\vartheta_1, \vartheta_2))$ ($q_F \in F, \vartheta_1, \vartheta_2 \in \{0, 1\}$) のとき、
 $\delta'(q_{ijk}, (a, b)) \Rightarrow (q_F', (\vartheta_1, \vartheta_2))$ (a_j, b_k の位置に $\$$ があれば、 δ' の対応する位置に $\$$ を入る)
- $W' = \{(a^\alpha \$, b^{\beta f(n)} \$)\} (n \geq 1)$ とする。

(補題 4) $\mathcal{L}(\mathcal{G}) \cap W = \emptyset$ と $\mathcal{L}(\mathcal{G}') \cap W' = \emptyset$ とは同値である。

(証明) 構成法より明らか。 \blacksquare

補題 4 に より $\Sigma = \{a, b, \$\}$, $W = \{(a^\alpha \$, b^{\beta f(n)} \$)\} (n \geq 1)$ である 2-TFA と W についてのみ考察すればよい。以下では、2-TFA

及び W はこのようなものとする。

(定義12) 2-TFA のある計算状況 (q, r, s) において, $(q, r, s) \neq (q, r+u, s+v)$ 且つ この列の中には同じ状態を持つものが両端の q を除いては現れないとき, この列をループといい, このループを $lp(Q, u, v)$ (Q : ループの中に現れる状態の集合) で表現する。 ■

(定義13) 2-TFA の計算状況の列 $(q_{p1}, r_{p1}, s_{p1}) \neq (q, r, s) \neq (q_{p2}, r_{p2}, s_{p2})$ に対して, ループ $(q, r, s) \neq (q, r+u, s+v)$ を挿入するとは, 計算状況の列 $(q_{p1}, r_{p1}, s_{p1}) \neq (q, r, s) \neq (q, r+u, s+v) \neq (q_{p2}, r_{p2}+u, s_{p2}+v)$ を作ることをいう。逆の操作をループの除去という。 ■

(補題5) 任意の 2-TFA $\mathcal{A} = (K, \Sigma, \delta, q_0, F)$ と W に対して, $\mathcal{L}(\mathcal{A}) \cap W \neq \emptyset$ ならば, 以下の条件を満たす $i, j \leq |K|$ が存在する。
 (1) $\mathcal{L}(\mathcal{A}) \ni (a^i c, b^j d)$

(2) $c = a$ 又は $\$, d = b$ 又は $\$$ で $c = d = \$$ ではない。

(証明) $\mathcal{L}(\mathcal{A}) \cap W \neq \emptyset$ だから i そのような W の元 $(a^{\alpha N} c, b^{\beta f(N)})$ とする。受理に至る計算状況の列を, $(q_0, 1, 1) \neq (q_F, r, s) (q_F \in F)$ とする。

(i) $t \leq |K|$ のとき, $w = (a^{r-1} c, b^{s-1} d)$ ($\alpha N = r-1$ のとき, $c = \$$, $\beta f(N) = s-1$ のとき $d = \$$ とする) とすれば, w は条件を満たす。

iii) $i \geq |K| + 1$ のとき (図2 参照)

この計算状況の列の中にループが存在する。すなわち、

$$(q_0, 1, 1) \vdash^i (q, j, k) \vdash^l (q, j+u, k+v) \vdash^m (q_F, r, s)$$

列よりループを除去した計算状況の列

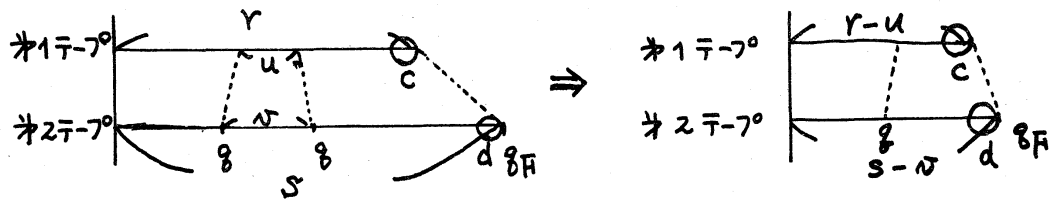
$$(q_0, 1, 1) \vdash^i (q, j, k) \vdash^m (q_F, r-u, s-v) \text{ を作る.}$$

$i+m \geq |K| + 1$ であれば、この中に再びループを見出すこと
が可能である。この操作を繰り返した結果得た計算状況の列を

$$(q_0, 1, 1) \vdash^n (q_F, p, q) \text{ とする. } (n \leq |K|) \text{ のとき,}$$

$$w = (a^{p-1}c, b^{q-1}d) \text{ (} dN = r-1 \text{ のとき, } c = \$, \beta f(N) = s-1$$

のとき, $d = \$$ とする) とすれば、 w はこの計算状況の推移
に従って受理され、 $p-1, q-1 \leq |K|$ である。 ■



(図2) 最終状態に至る計算状況の列からのループ
の除去

補題5によつて、言語 $(a^i c, b^j d) \ (i, j \leq |K|)$ で $L(5)$ の元であ
るものが存在することである。 $L(5) \cap W \neq \emptyset$ の必要条件であるこ
とが、 $L(5)$ の元である。すなわち、言語 $(a^i c, b^j d)$ の受理に至る計算
状況の列に、許される適当なループを挿入したとき、ある n

に対して $(a^n, b^{f(n)})$ の受理に至る計算状況の列とすることから出来るか否か』を問題 A と書くとき、問題 A が可解であれば、 $(S) \cap W = \emptyset$ か否かは可解となる。

(注) 問題 A で使用されるループの種類は、定義により、2-TFA より有限個に定まる。これらのループを $lp_i(Q_i, \alpha_i, \beta_i)$ で示し、その集合を LP で示す。

(補題 6) 問題 A は可解である。

証明の中で使用する補題を示す。

(補題 7) $\forall \alpha_i (1 \leq i \leq l)$: 正の整数, $\forall p$: 正の値をとる多項式が与えられたとき,

$$\sum_{i=1}^l \alpha_i \eta_i = p(n)$$

を満たす整数 $n, \eta_i (1 \leq i \leq l)$ が存在するか否かは、決定可能である。

(証明) $G = \text{G.C.D.}(\alpha_1, \dots, \alpha_l)$ とすると $\sum \alpha_i \eta_i = Gt$ といえる。よって、 $p(n) \bmod G = 0$ となる n が存在するか否かを決定すればよい。 $p(n) \bmod G = p(n \bmod G) \bmod G$ であり、この問題は決定可能である。 \square

(補題 8) $\forall \alpha_i (1 \leq i \leq l)$: 正の整数が与えられているとき,

$$\sum_{i=1}^l \alpha_i \eta_i = n$$

において、 n を大きくするとき、 $\forall i$ に対して定数 C_{ij} が存在して $0 < \eta_j < C_{ij} (i \neq j)$ となる整数解 $\eta_i (1 \leq i \leq l)$ が存在する。 ($\text{G.C.D.}(\alpha_1, \dots, \alpha_l) = 1$)

(証明) 先ず, $\forall \alpha, \beta$ ($\text{G.C.D}(\alpha, \beta) = 1$) に対し n は 1 より大きく $n < \alpha + \beta$ と, ても, $\alpha x + \beta y = n$ には $0 < y < c$ (c : 定数) とする解 (x, y) を持つことを示す.

$\alpha x + \beta y = 1$ の解 (x_0, y_0) とする. $m = \min(\alpha, \beta)$ とする. すると n は 1 より大きく $n < \alpha + \beta$ とする.

$n = L\alpha + (m|y_0| + 1)(\alpha + \beta) + Mm + i$ ($1 \leq i \leq m, 0 \leq Mm < \alpha$) と書ける. $\alpha(i x_0) + \beta(i y_0) = i$ であるので,

$$n = (L + m|y_0| + 1 + i x_0)\alpha + (m|y_0| + 1 + i y_0)\beta + Mm$$

よって, $0 < y < m|y_0| + m y_0 + M + 1$ とする解が存在する.

さて, $\forall i$ に対して,

$$\begin{aligned} d_1 \eta_1 + \dots + d_{i-1} \eta_{i-1} + d_{i+1} \eta_{i+1} + \dots + d_e \eta_e + d_i \eta_i \\ = d_1 \eta_1 + \text{G.C.D}(d_2, d_3, \dots, d_{i-1}, d_{i+1}, \dots, d_e, d_i) \eta \end{aligned}$$

とすると, $0 < \eta_1 < C_{i1}$ なる解が存在する. $i = 2$

$$\alpha_j / \text{G.C.D}(d_2, d_3, \dots, d_i) = d'_j \text{ とすると}$$

$$\alpha'_2 \eta_2 + \dots + \alpha'_i \eta_{i-1} + \alpha'_{i+1} \eta_{i+1} + \dots + \alpha'_e \eta_e + \alpha'_i \eta_i = n$$

同様のことを繰り返すと $\forall j \neq i$ に対して $0 < \eta_j < C_{ij}$ となる解がある. \square

(補題 6 の略証)

(i) $c = a, d = b$ の場合

問題 A が可解となることは明らかである.

④ c または d のいずれ b かつ $\$$ である場合

$(s_1, s_2) = (a^i c, b^j d)$ の受理に至る計算状況の列を

$(q_0, 1, 1) \leq (q_F, i+1, j+1)$ と書く.

このとき, 計算状況の列に挿入可能なループの集合を求めるアルゴリズムを与える.

(Algorithm IR)

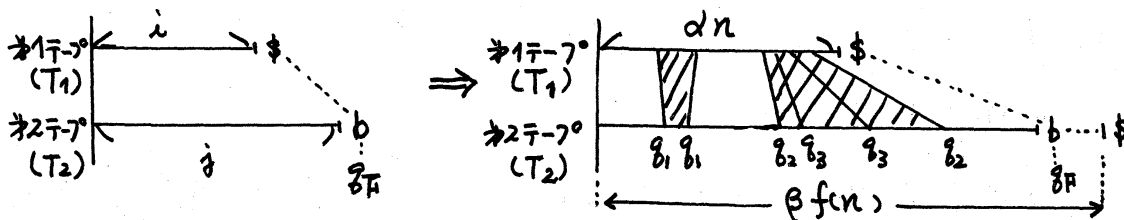
step 1) 計算状況の列中に現われた q_F 以外の状態の集合を Q , $L = \emptyset$ とする.

step 2) LP 中の $\forall i$ に対して $Q_i \cap Q \neq \emptyset$ ならば $Q = Q \cup Q_i$, $L = L \cup \{lp_i\}$

step 3) step 2 を L が増えなくなるまで繰り返す. ■

(a) $\deg(f(n)) > 1$ の場合

(1) $c = \$$ のとき. (図3参照)



(図3) ループの挿入

図3のように T_1, T_2 にいくつかのループを挿入して, T_1 の長さ $\alpha n + 1$, T_2 の長さは $\beta f(n)$ 以下となるようにすることによって出来るか否かを決定すればよい. Algorithm IR の出力 L を $\{lp_1, \dots, lp_p\}$ とするとき, 問題は,

$$\left\{ \begin{array}{l} i + \sum_{k=1}^p \alpha_k \eta_k = \alpha n \quad (1) \\ j + \sum_{k=1}^p \beta_k \eta_k < \beta f(n) \quad (2) \end{array} \right.$$

い. ここで、 $\eta_k = 0$ のとき、間接的に $\eta_k = 0$ でなければなら
 ないもの p が存在する p が、補題 7.8 より (1) の解 n は非負に無数
 に存在し、 $1 \leq p$ として、自然数解 η_k を与える。逆に、いくつ
 かの η_k を 0 に限定しても、無限定で (1) に解 p がなければ、解
 を有さないめで、(1), (2) の自然数解を考察すればよい。

補題 7.8 f が (1) をみたす, 自然数解 η_k の存在は決定可能であつた. $\max_k [\beta_k / \alpha_k] = M$ とおく.

$$\sum_k \eta_k \beta_k < M \sum_k \eta_k \alpha_k \text{ である. } \delta, \tau,$$

$$j + \sum_k \beta_k \eta_k < M_i + M \sum_k \alpha_k \eta_k + j - M_i = M \alpha_n + (j - M_i) .$$

$\deg(f(n)) > 1$ かつ $Mdn + (j - M_i) < \deg(f(n))$ とする n に対して n が 4 に
 等しいとき n が 4 に等しい。

(□) $d = \$ 9 \text{ と } 5$

(イ)と同様にして、

$$\begin{cases} i + \sum_{k=1}^p \alpha_k \eta_k < \alpha n & (3) \\ j + \sum_{k=1}^p \beta_k \eta_k = \beta f(n) & (4) \end{cases}$$

51). (4) 与えた可自然数 n, m が存在する p を p_1 とし、補題 7.8 より決定可能である。

$$(*) \min_k [\alpha_k / \beta_k] = m > 0 \quad \alpha \in \Xi$$

$$\sum \alpha_k \eta_k \geq m \sum \beta_k \eta_k \quad \text{for } \eta.$$

$\alpha n > i + \sum \alpha_k \eta_k \geq m \sum \beta_k \eta_k + m j + i - m j = m \beta f(n) + (i - m j)$
 (1) $K \not\models \exists x, \deg(f(x)) > 1$ であり n に上限 N が存在する。 (2) $K \models$
 して、可解となる。

(**) $\min_k [\alpha_k / \beta_k] = 0$ とならば $\exists t$ s.t. $\alpha_t = 0$ となる。

(*) の解において n を大きくと、ても、補題 8 より、 $\alpha_k = 0$
 となる t に対して $t \neq k$ で $0 < \eta_k < C + k$ となる定数 C が
 存在する。そのように解 η_k をとると、

$$i + \sum \alpha_k \eta_k = i + \sum_{k \neq t} \alpha_k \eta_k < C \quad (C: \text{定数})$$

となるので解は存在する。

以上より、この問題は可解となる。

$\deg(f(n)) \leq 1$ の場合も同様にして可解であることが証明で
 きる。 ■

(定理) 任意の 2-TFA $\mathcal{G} = (K, \Sigma, \delta, q_0, F)$ ($\Sigma = \{a, b, \$\}$) と W
 $= \{(\alpha^n \$, b^{ef(n)} \$) \mid (n \geq 1, f: \text{多項式})\}$ に対して、

$\mathcal{L}(\mathcal{G}) \cap W = \emptyset$ ならば可解である。

(証明) 補題 5.6 による。 ■

(系) 図 1 の PPS に関する上述のデッドロック問題及び相互
 排他問題は可解である。 ■

4. 結び

本報告で示した結果は次のようにいくら拡張できる。例

へば、定理は任意の α の W に対して有効であることから、このような W に変換可能であるような 2-TFA と語の集合に対しては同じ議論が成立する。したがって、 $W = \{(a_1 \dots a_k) (a_{k+1} \dots a_j)^n, (b_1 \dots b_k) (b_{k+1} \dots b_m)^{f(n)}\}$ となる PPS に対しても、ここでとり上げたデッドロックと相互排他問題は可解となる。プロセスが3以上、又プロセスの形が複雑な場合については検討中である。

謝辞 日頃、御指導をいただく本学福村晃夫教授及び熱心に御討論くださる研究室の皆様に感謝します。

文献

- (1) HANSEN, P.B., Concurrent-PASCAL Report, Information Science, California Institute of Technology, NSF QCA DCR 74 1733/CPI (1975)
- (2) KARP, R. & MILLER, R., Parallel Program Schemata, JCSS (1969)
- (3) ASHCROFT, E. & MANNA, Z., Formalization of Properties of Parallel Programs, Machine Intelligence (1971)
- (4) 伊藤貴康, プログラム理論, コロナ社 (1975)